Incremental Object Discovery in Time-Varying Image Collections

Theodora Kontogianni Markus Mathias Bastian Leibe
Visual Computing Institute, Computer Vision Group
RWTH Aachen University
{kontogianni, mathias, leibe}@vision.rwth-aachen.de

Abstract

In this paper, we address the problem of object discovery in time-varying, large-scale image collections. A core part of our approach is a novel Limited Horizon Minimum Spanning Tree (LH-MST) structure that closely approximates the Minimum Spanning Tree at a small fraction of the latter's computational cost. Our proposed tree structure can be created in a local neighborhood of the matching graph during image retrieval and can be efficiently updated whenever the image database is extended. We show how the LH-MST can be used within both single-link hierarchical agglomerative clustering and the Iconoid Shift framework for object discovery in image collections, resulting in significant efficiency gains and making both approaches capable of incremental clustering with online updates. We evaluate our approach on a dataset of 500k images from the city of Paris and compare its results to the batch version of both clustering algorithms.

1. Introduction

Social media platforms have become favorite storage and sharing sites for all kinds of images. A large part of those images are touristic photos, resulting in a dense image coverage of famous monuments and landmarks up to the scale of entire cities [8, 2]. This has created a call for computer vision algorithms that can perform efficient object discovery [16, 15, 22], clustering, and matching in image collections for applications such as large-scale 3D reconstruction [13, 1, 7], scene summarization [19], automatic image annotation [8], or visual search [24, 2].

However, the content of large-scale image repositories is never static. Millions of images are added to such repositories each day, while others are withdrawn or deleted. Current object discovery algorithms [16, 15, 24, 2, 22] do not yet address this issue. They typically operate in a static setting, making it necessary to re-run the entire clustering process whenever the underlying image database changes, even though only a small part of the clusters may be affected by the changes. Image retrieval and recognition web services using those algorithms waste thousands of core hours of computation because of this.

Even leading providers of visual search engines such as Google are bound to rebuild the clustering from scratch every week [personal communication] due to changes in the image database. Currently this problem of dynamic database changes is not well researched in the community despite its large effect on practical applications.

In this paper, we address the image clustering and object discovery problem in an incremental setting. We propose a novel clustering method that allows for efficient reuse of the stored data. A key idea of this paper is that many clustering methods can be efficiently implemented with the help of a spanning tree. In order to enable incremental clustering, we therefore propose efficient techniques for incrementally constructing and updating the spanning tree. Because of the incremental updates, information about unaffected clusters can be preserved and only those clusters need to be recomputed that contain updated parts of the spanning tree.

At the core of our approach is a novel Limited Horizon Minimum Spanning Tree (LH-MST) data structure that closely approximates a Minimum Spanning Tree (MST), while being significantly faster to construct and to update whenever new images are added to the matching graph. The LH-MST can be used in any object discovery approach that operates on spanning trees, such as single-link clustering [16, 8, 24] or Iconoid-Shift mode estimation [22, 23]. In contrast to a regular MST that always requires the full matching graph to be known a priori, an LH-MST can be computed starting from a local seed, making it suitable for local exploration of a matching graph through Query Expansion [5]. We experimentally verify that in realistic use cases, the differences to a regular MST are minimal, making our algorithm an efficient alternative for parallel and distributed clustering applications.

We demonstrate our approach's practical feasibility for large-scale incremental object discovery by applying it to the *Iconoid Shift* (IS) object discovery approach [22, 23]. As our results will show, replacing the MST by an LH-MST translates to a 5-fold improvement in runtime already during batch operation. In addition, we experimentally show that IS with LH-MST achieves stable incremental clustering results even when the image database is extended by a significant fraction.

In detail, this paper makes the following contributions. (1) We propose a novel LH-MST spanning tree structure, which approximates the MST and which can be incrementally updated very efficiently. (2) We show how this tree structure can be used for efficient approximative single-link clustering and how it can be incorporated into the Iconoid Shift approach [22] for large-scale object discovery in image collections. (3) We verify experimentally that the resulting IS using LH-MST (or LH-IS) approach achieves very similar results as the offline version of IS, while being both considerably faster and capable of online updates of a significant fraction of the image database.

Related Work. Object discovery approaches aim at finding clusters of images showing the same object or landmark building in large image collections. The standard procedure for this is to apply local feature based matching techniques [14] to build up a matching graph of images and to then subdivide the connected components of this matching graph into image clusters. A large range of clustering methods have been proposed for this step, including single-link hierarchical agglomerative clustering [16, 8, 24], spectral clustering [15, 12, 9], eigenvector centrality [10], spherical k-means [19], Kernel Vector Quantization [2] and Medoid Shift mode estimation [22, 23]. Because of the large size of the datasets to be mined, runtime efficiency and ease of parallelization are a prime concern.

In this paper, we focus on two classes of object discovery algorithms that have proven their worth in large-scale applications: single-link clustering [16, 8, 24] and Iconoid Shift [22, 23]. Single-link clustering is directly related to the Minimum Spanning Tree (MST). Given the MST of a graph, the corresponding single-link clusters can be obtained by cutting all MST edges above a chosen distance threshold [17]. Iconoid Shift [22] uses Medoid Shift mode estimation with a special kernel in order to find images that have a locally maximal mutual homography overlap with their neighbors. Those images often correspond to central, iconic views of a landmark building, and the overlapping images under the kernel form the resulting cluster. In order to be able to apply the Medoid Shift formalism, the used distance measure needs to fulfill the triangle inequality. Since this is usually not the case for local-feature based matching scores, IS defines a transitive pairwise image distance measure that is computed over the edges of an MST. In practice, the MST construction step is responsible for a significant fraction of both clustering algorithms' runtime.

As shown in [20, 3], the MST of a graph with N nodes can be updated in $O(N)$ when a new node is added to the graph, which is simultaneously the lower bound for an exact update. This may not seem like much, but the key issue here is that such an exact update requires the full matching graph to be available on a single machine. For practical object discovery in large-scale image databases, even individual connected components of the matching graph may get so large that processing them on a single computing node becomes inefficient (in our experiments with 500k images, the largest connected component comprises > 50k images). We therefore aim at an approximation of the MST that can be locally grown from a seed image, such that computation can be distributed over a computing cluster.

2. Clustering Methods for Object Discovery

In this paper we are primarily interested in performing efficient object discovery in large image collections. To that end, we propose a novel tree structure which can speed-up MST based clustering algorithms and has the additional benefit of allowing incremental database updates without the need of rebuilding the entire database. In the following we will review two such clustering algorithms which rely on the creation of an MST. In the remainder of the paper we will then show that these algorithms benefit from our proposed MST approximation.

2.1. Single-Link Agglomerative Clustering

Single-Link Agglomerative Clustering is a widely used clustering method especially in the landmark discovery community [16, 8, 24]. It offers the advantage of creating clusters without any assumptions on their shape and the number of clusters. One of its drawbacks is that it is prone to long chain-like clusters that require rigorous post-processing in order to achieve competitive results. Additionally it does not allow for incremental database updates, requiring a full re-computation after adding nodes.

Single-Link clustering can be performed in a bottom-up fashion with each image starting as an individual cluster. Then the two closest clusters are iteratively joined until either the whole database is represented by a single cluster or a cut-off threshold θ on the between-cluster distance $d(A, B)$ is reached. This cluster distance is defined as $d(A, B) = \min_{a \in A, b \in B} d(a, b)$.

An efficient O(N^2) implementation of Single-Link clustering can be obtained by first building the full matching graph and then computing its MST. Removing all edges above a certain threshold θ generates clusters of connected components. This dependency on the MST creation makes it an ideal candidate for MST approximation experiments.

2.2. Iconoid Shift

Iconoid Shift (IS) discovers landmark objects in image collections by searching for so-called *iconic* images. Starting from a set of seed images (typically obtained by Ge-

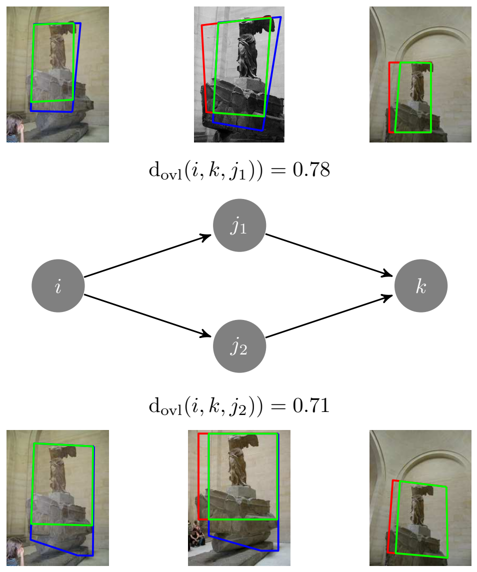

$\mathrm{d}_{\mathrm{ovl}}(i,k,j_1)) = 0.78$
j_1
i
k
j_2
$\mathrm{d}_{\mathrm{ovl}}(i,k,j_2)) = 0.71$

Figure 1: Transitive overlap distance

ometric MinHash [4]), IS first builds up local matching graphs G by recursively applying image retrieval [14, 5] with the seed image r as a query. The edges between images i and j are weighted by a distance measure $\mathrm{d}_{\mathrm{ovl}}(i,j)$ that quantifies the overlap region between the two directly connected images. In addition to those direct distances, IS proposes a transitive overlap distance $\mathrm{t}_{\mathrm{ovl}}(i,j)$ that propagates content overlaps between images that are not directly connected via homography chains. The algorithm adds nodes to G as long as the transitive distance to the root node r is below a cut-off point β (the so-called kernel radius). As a result of this local exploration, each seed image is connected to a set of similar images via the graph structure G.

On this graph, IS first builds up an MST and then computes the transitive distances $\mathrm{t}_{\mathrm{ovl}}(i,j)$ between all pairs of nodes (i,j) in G by propagating homography overlaps via the MST. It then uses the *Medoid Shift* (MS) algorithm [18] to find the modes of the density implicitly defined by the transitive overlap distance, which correspond to the *iconic* images. Medoid Shift is a generalization of the Mean Shift mode estimation algorithm [6] that operates on graphs. In each iteration, it computes the medoid of the graph nodes under its kernel (defined by the pairwise image distances) and then shifts the kernel window to this medoid. Whenever such a shift happens, the local matching graph is extended using the medoid as a new seed and the MST is recomputed. It iterates until convergence (see Algorithm 1).

Transitive overlap distance ($\mathrm{t}_{\mathrm{ovl}}$): The transitive overlap distance forms an important part of calculating the overlap over paths in the graph. It defines a distance for a pair of images (i,k) which are not directly connected in G, only through a path j of nodes. All directly connected images in G have corresponding homographies and bounding boxes enclosing the matching feature points. As shown in Figure 1, the transitive overlap distance between images i and k through images j_1 or j_2 is computed by projecting the matching bounding box from images i (blue box) and k (red box) to their common neighbor j_1 or j_2. The overlap region between these two projections is found by intersecting the boxes (green box in j_1 and j_2) and back-projecting them to images i and k. Then the transitive overlap distance $\mathrm{t}_{\mathrm{ovl}}(i,k,j)$ is defined as 1 minus the minimum overlap of the projected boxes divided by the image area (see [22]).

Figure 1 also shows that the path j influences the final distance between i and k, i.e. the overlap via j_1 is much larger than via j_2. There are as many distances as paths from i to k. The choice of the spanning tree can therefore crucially influence the behavior of IS.

Caveats. The use of MS in a distance graph G requires the edge weights to form a metric. However due to the limited repeatability of the features used for image retrieval, the graph G may contain cycles which violate the triangular inequality for d_{ovl} in G, i.e. $\mathrm{d}_{\mathrm{ovl}}(i,k) \not\leq \mathrm{d}_{\mathrm{ovl}}(i,j) + \mathrm{d}_{\mathrm{ovl}}(j,k)$. Although d_{ovl} is symmetric, the retrieval system is not. As such, it is not possible to use MS directly on G (at this point G is not fully connected). The spanning tree construction is necessary in order to circumvent this problem. By computing the MST on G, all potentially inconsistent cycles are removed, before the fully-connected, consistent graph G' is constructed by connecting nodes via transitive connections $\mathrm{t}_{\mathrm{ovl}}$ (see Algorithm 1). Through the definition of $\mathrm{t}_{\mathrm{ovl}}$ (see [22]), the resulting graph fulfills the triangular inequality, making it possible to perform an iteration of MS.

When the mode shifts to a new node, the retrieval has to be performed again, as new nodes might now be within reach of the kernel distance, while others might drop out. This means that the MST has to be recomputed on the new local neighborhood. Although this new neighborhood is usually smaller than the full connected component, this step can still incur heavy computation. An efficient alternative to the MST creation can therefore result in significant speedups.

In summary, we see that, as for single-link clustering, the expensive calculation of the MST ($\mathrm{O}(N^2)$ in the number of tree nodes N) is a major bottleneck of IS and it restricts its usability as an online algorithm.

3. Limited Horizon Minimum Spanning Tree

In this section we introduce a novel tree structure called ***Limited Horizon Minimum Spanning Tree*** (LH-MST). This tree structure (Section 3.1) is built during image retrieval (e.g. in Step 1 in Algorithm 1). It is considerably faster to compute than the regular MST (Section 3.1.1) and allows to incrementally add new nodes to an initial dataset instead of rebuilding the whole database when new imagery

Algorithm 1: Iconoid Shift Steps

```
Require: photo collection $C$, seed image $s$
  while $s$ shifts to a new mode do
    Step 1. Build graph $G$ around $s$ that connects similar images via distance $\mathrm{d_{ovl}}$.
    $\rightarrow$ Edge weights are not a metric.
    Step 2. Create minimum spanning tree $T$ of $G$ to remove cycles.
    Step 3. Construct fully connected $G'$ from $T$ by replacing missing edges with $\mathrm{t_{ovl}}$.
    $\rightarrow$ Edge weights are now a metric.
    Step 4. Perform Medoid Shift in $G'$. Mode $m$ is now the new $s$.
```

Algorithm 2: Compute LH-MST

```
Require: root $r$, priority queue $\mathrm{q}_{update}$
  $\mathrm{q}_{update} = \emptyset$
  $push(\mathrm{q}_{update}, \{\mathrm{edge}(r, r), \mathrm{d_{ovl}}(r, r)\})$
  while $\mathrm{q}_{update} \neq \emptyset$ do
    $\mathrm{edge}(p, n) = pop(\mathrm{q}_{update})$
    if $n \notin$ spanning tree $T$ then
      $add(\mathrm{edge}(p, n))$ to $T$;
      $children = retrieve(n)$
      for $c \in children$ do
        if $\phi(\mathrm{t_{ovl}}(r, c)) < \theta$ then
          $push(\mathrm{q}_{update}, \{\mathrm{edge}(n, c), \mathrm{d_{ovl}}(n, c)\})$
  return $T$
```

is available.

3.1. Offline Construction of LH-MST

Algorithm 2 shows the creation of our ***Limited Horizon Minimum Spanning Tree*** (LH-MST). The algorithm is based on the priority queue q_{update}, which contains tuples of edges and their corresponding edge cost $\mathrm{d_{ovl}}$. The edges are directed from a parent node p to a child node n and the queue is sorted by the edge cost in ascending order. As before, we start with a seed, the root image r. The priority queue is initialized by a directed edge from $p = r$ to $n = r$ with weight equal to one. As long as the priority queue contains elements, we remove the first tuple $\{\mathrm{edge}(p, n), \mathrm{d_{ovl}}(p, n)\}$ from q_{update} and perform image retrieval on image n. Newly retrieved images c are added to the q_{update} (as $\{\mathrm{edge}(n, c), \mathrm{d_{ovl}}(n, c)\}$) if they are within kernel distance to the root node and do not create cycles. A few iterations of the algorithm are visualized in Figure 2.

LH-MST is an MST approximation constructed in a greedy way. After iteration k the resulting spanning tree contains at most k nodes and has a maximal depth of k. The resulting spanning tree would be equivalent to the MST if the retrieval system were symmetric.

3.1.1 Complexity Analysis.

A major advantage of using LH-MST over MST to enforce the triangular inequality in G is speed. The LH-MST is constructed during retrieval and causes negligible maintenance overhead during the update. **Step 2** of the IS pipeline (see Algorithm 1) is not needed anymore. Saving the overhead for the MST construction ($\mathrm{O}(N^2)$) makes the proposed method significantly faster.

In contrast, one might suggest to use the basic IS algorithm as is and only replace the MST creation by a faster approximate MST algorithm [11]. The resulting speed-up would be limited, as any faster MST algorithm still needs to build a full image graph first (not a spanning tree) and then compute the MST as a separate step.

Algorithm 3: Incremental Update LH-MST- Addition

```
Require: root $r$, priority queue $\mathrm{q}_{update}$
Require: spanning tree $T$
  $\mathrm{q}_{update} = \emptyset$
  for $\mathrm{edge}(p, n) \in T$ do
    $push(\mathrm{q}_{update}, \{\mathrm{edge}(p, n), \mathrm{d_{ovl}}(p, n)\})$
  while $\mathrm{q}_{update} \neq \emptyset$ do
    $\mathrm{edge}(p, n) = pop(\mathrm{q}_{update})$
    if $n \notin T$ then
      $add(\mathrm{edge}(p, n))$ to $T$
      $children = retrieve(n)$
      for $c \in children$ do
        if $\phi(\mathrm{t_{ovl}}(\mathrm{r}, \mathrm{c})) > \theta$ then
          $push(\mathrm{q}_{update}, \{\mathrm{edge}(n, c), \mathrm{d_{ovl}}(n, c)\})$
  return T
```

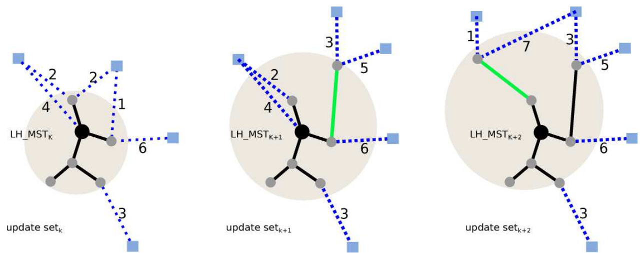


Figure 2: Three Steps of the Limited Horizon Minimum Spanning Tree. **LH-MST** $_k$**:** The current tree (solid lines) contains k nodes, the update set new node candidates with their corresponding weights (dashed lines). **LH-MST** $_{k+1}$**:** the node with the minimal distance to any existing tree node (green line) is added to the tree; redundant edges are removed; the retrieval system finds new image candidates (within kernel distance) and adds them to **update set**$_{k+1}$.

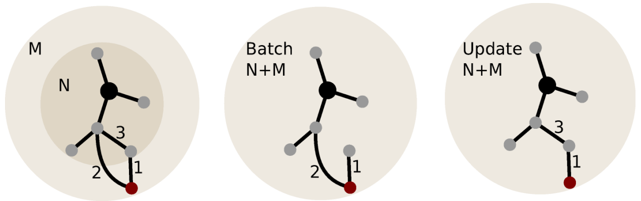


Figure 3: The incremental update of the LH-MST can result in slightly different trees. The red node belongs to set M, all other nodes to set N. In batch mode (middle), the red node would be retrieved during tree-construction with cost “2”. The incremental update (right) starts with a pre-existing tree resulting in a slightly different path.

3.2. Incremental Update of LH-MST

Performing any kind of clustering on databases with hundreds of thousands of images is very time consuming. Although we are able to speed up the tree creation, it might well happen that the image database has already been outdated before the clustering finished. Instead of collecting new data and re-running the clustering, the LH-MST is able to perform incremental updates. In this section we will describe the update procedure; Section 4.3 will then experimentally show its validity.

Algorithm 3 shows the incremental update of an LH-MST. The algorithm extends an already existing spanning tree LH-MST $^{(N)}$ consisting of N nodes with a set of new nodes M to create the spanning tree LH-MST $^{(N+M)}$. The update procedure is similar to the initial construction of an LH-MST. Instead of starting from the root node r, we start with a complete tree built based on set N, and fill the priority queue with edges from this initial tree. The retrieval is performed in the order of the priority queue, but only retrieves images from the new set M. This adds new edges to the existing tree from N to M. As such, the resulting trees might slightly differ, as shown in Figure 3.

3.2.1 Complexity Analysis

We consider the worst case scenario (in terms of complexity) of a fully connected retrieval graph, where each node upon retrieval returns all other nodes in the connected component. As complexity we consider the number of distance measure computations between nodes. Our full database has N elements.

Offline Version considers the construction of an LH-MST consisting of N nodes (including the seed s). In iteration k=1, there is only one node already in the LH-MST, the seed s and N-1 nodes to be considered for addition. Thus, N-1 distance computations between node pairs are needed. Each iteration of adding a node to the tree reduces the number of distance computations by one, resulting in a complexity of

$$\text{compl}_{\text{offline}} = 1+2+3+4+\cdots+N-1 \quad (1)$$
$$= \frac{(N)\cdot(N-1)}{2} \quad (2)$$

Online Update considers a pre-existing LH-MST with an initial set of nodes N_1 and updates the initial tree with a new set of nodes N_2, where $N_1+N_2=N$. In iteration k=1 there are $N_1 \cdot N_2$ distances computed between all the N_1 nodes of the initial spanning tree and all the N_2 nodes of the update set in order to pick a new edge to add in the spanning tree. Similar to the offline update, in each iteration elements from N_2 will be added to the tree resulting in a decrement of distance computations:

$$\text{compl}_{\text{update}} = N_1 \cdot N_2 + \frac{(N_2)\cdot(N_2-1)}{2} \quad (3)$$

As expected, the complexity of the offline version is larger than the complexity of incrementally updating the existing initial tree with the additional nodes by

$$\text{compl}_{\text{offline}} - \text{compl}_{\text{update}} = \frac{(N_1)\cdot(N_1-1)}{2}, \quad (4)$$

which is the number of distance computations to create the spanning tree with the initial set of nodes N_1. Note that this analysis includes the effort for retrieval and distance computations. Without considering retrieval, the remaining effort of node addition is in O(1).

4. Experiments

In this paper we propose to replace MST by LH-MST in order to speed up computation and allow for efficient online updates. We will now use LH-MST to perform Single-Link Agglomerative clustering and Iconoid Shift as follows.

Limited Horizon Single-Link Clustering (LH-SL). In the tree construction we replace MST by LH-MST.

Limited Horizon Iconoid Shift (LH-IS). We replace steps 1 and 2 in Algorithm 1 by creating the tree T' directly.

LH-MST is one instance of many possible spanning tree alternatives that would also allow for fast creation and online updates. The alternatives could also be constructed during the retrieval step (similar to LH-MST) and thus would be faster than the original MST. We will show the advantages of LH-MST over several of such spanning tree baselines: **BFT** - *Breadth First Spanning Tree* (siblings with higher priority in queue), **DFT** - *Depth First Spanning Tree* (children with higher priority in queue), **RAND** - *Random Spanning Tree* (random order priority queue), **SPT** - *Shortest Path Spanning Tree* (queue sorted by shortest path to initial medoid).

All experiments are performed on the full 500k images of the Paris dataset [21] collected from Flickr and Panoramio.

4.1. Evaluation of Single-Link Clustering

In Single-Link Clustering changing the cut-off threshold θ affects the number of resulting clusters starting from one cluster for the whole database to individual clusters for each element. Replacing the MST creation with LH-MST will likely result in a slightly different tree structure and therefore in different clusters. To assess the magnitude of this difference we compare the resulting trees by their edit-distance E_d as a function of θ:

$$E_d(S,T) = \tfrac{1}{2}(|\{(i,j)|(i,j)\in S \wedge (i,j)\notin T\}| + |\{(i,j)|(i,j)\notin S \wedge (i,j)\in T\}|)$$

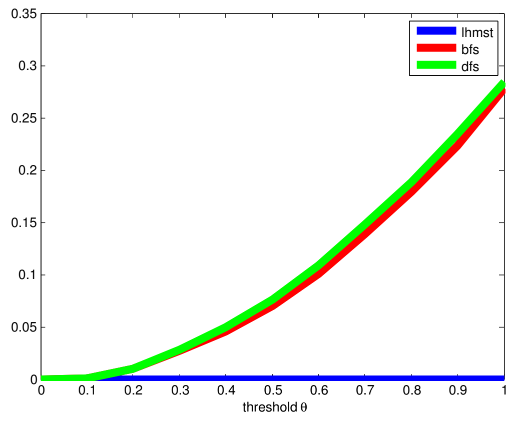


Figure 4: The cumulative histogram of image distances in the cases of disagreeing tree edges.

where T is the MST and S is the candidate tree. First the two full trees are compared based on the number of edges that need to be removed/added to convert the approximation (LH-MST) back into MST. We then keep track of the tree differences when changing θ.

Figure 4 shows the percentage of changed edges in the alternative trees (BFS,DFS,RAN,LH-MST) compared to the MST when varying θ. The LH-MST is the closest to the MST by a large margin.

While this result shows that the approximation does not deviate by more than 0.027 percent for the original MST, it should be noted that the performed comparison is still slightly too strict. From a practical point of view, the underlying tree structure is irrelevant as long as the same images are clustered together by both methods.

4.2. Evaluation of Offline LH-IS

After using LH-MST for the case of Agglomerative Clustering, we now show its validity of performing Iconoid Shift with LH-MST called LH-IS. We compare our results to the ones of [22]. In the following, we introduce the general setup and evaluation measures used for the evaluation.

Seeds. We use the seeds provided by the authors of [22]: A seed group S with geometric min-hash [5] using 5 min-Hash sketches of size 2 returns a total 10487 images. Duplicates are removed resulting in a set of 477 images. Empirically this limited number of seeds offers a good coverage of the database.

Medoid Shift kernel. We used a hinge kernel for the MS with a cut-off threshold of 0.9, which means that we allow images with at least 10% overlap to the support set of the $Iconoid$.

Evaluation measures glossary. In the following we compare various evaluation measures to the “ground truth”, the set of medoids $\mathbb{M}^{MST}$ originating from the original (offline) version of IS. In Section 4.3 we compare to the online version and use the sets $\mathbb{M}^{LH-IS}$ as “ground truth”.

τ	$\lvert\mathbb{M}\rvert$	$\lvert\mathbb{M}_A\rvert$	$\lvert\mathbb{M}_O\rvert$	$\lvert\mathbb{M}_U\rvert$	$\lvert\mathbb{K}\rvert$	*Rank* avg.	$\lvert\mathbb{S}_{sup}\rvert$ avg.	*DB cover*
IS	**324**	-	-	-	90	-	380.42	73249
LH-IS	321	**282**	28	4	87	18.71	380.43	70762
BFT-IS	396	150	213	14	153	172.6	1146.27	73886
DFT-IS	365	137	136	87	154	75.0	331.65	68023
SPT-IS	256	88	148	12	220	132.39	1334.86	61828
RAN-IS	346	194	135	15	99	61.29	981.7	70414

Table 1: Performance of the LH-MST compared to baselines when replacing the MST in the IS framework.

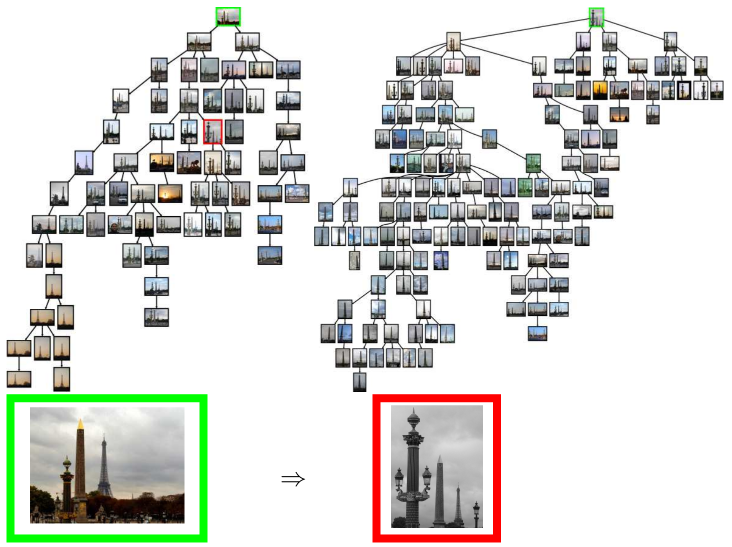
τ	potential spanning tree are $\tau \in \{$ MST, LH-MST, BFT, DFT, RAN, SPT $\}$
$\mathbb{M}^{\tau}$	the set of medoids resulting from method τ
$\mathbb{M}_A$	the set of medoids common between ground truth and method τ.
$\mathbb{M}_O$	the set of medoids of method τ that do not match the medoid selected in the ground truth. Still, the medoids are members of the ground truth cluster. How close they are to the ground truth is expressed by the *Rank*.
$\mathbb{M}_U$	the set of medoids of method τ that are not even part of any ground truth cluster.
$\mathbb{K}$	the set of seeds for which the MS is stuck in initialization.
$\mathbb{S}_{sup}$	the cardinality of the final clusters
Rank	the smallest relative position (rank) that the medoids of method τ achieve in all ground truth clusters. The smaller the value of $\mathbb{R}_d$ is, the closer the result is to the optimum.
DB Cover	the size of the database explored

Table 2: Glossary: Evaluation Measures

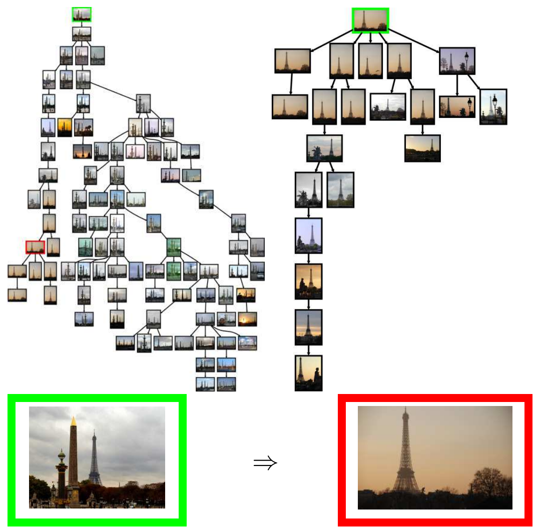
4.2.1 Results of Offline LH-MST

Table 1 summarizes the results of our evaluation of using different spanning trees in place of the MST. As it can be seen, using LH-IS provides the closest approximation of IS. The majority of clusters are represented with the same medoid (87% or 282 medoids). In 28 cases the algorithm converges to a different medoid which is part of the same cluster. Figures 7e and 7f show exemplary samples of the result of LH-IS and IS. Most picked medoids are the same, in case of divergence the images are still similar. This is also expressed by the low rank difference of 18.71. In 4 cases the medoid does not have a direct correspondence, i.e. the medoid is not part of the support set of MST. Such an example is shown in Figure 5.

Interestingly, the second most similar result to IS is achieved by RAN-IS, confirming that in principle any tree could perform well as long as it is not biased towards certain nodes. An important property of ’good’ spanning trees is to allow IS to shift away from its initial seed. SPT-IS is



(a) 2 iterations of LH-MST



(b) 2 iterations of MST

Figure 5: Example iterations of IS using (a) LH-MST and (b) MST. Both algorithms are initialized with the same seed image (green border) that depicts the *Eiffel Tower* and *Cleopatra's Needle* but they converge to different medoids (red border). The MST's final cluster (b) shows only one of the landmarks (*Eiffel Tower*). As can be seen, the final medoid of LH-MST does not belong to the support set of the final medoid MST and visa versa. This is one of the rare cases where medoids cannot be matched (see Table 1, column $\mathbb{M}_U$).

LH-MST			$\|\mathbb{M}\|$	$\|\mathbb{M}_A\|$	$\|\mathbb{M}_O\|$	$\|\mathbb{M}_U\|$	$\|\mathbb{S}\|$	*Rank*	$\|\mathbb{S}_{sup}\|$	*DB cover*
	Split									
	100	-	321	-	-	-	87	-	416	70762
S1	90	10	327.33 ± 1.89	263.0 ± 8.83	53.67 ± 8.96	3.67 ± 1.89	90.0 ± 2.16	79.93 ± 74.43	424.04 ± 32.72	61716
S1	70	30	327.0 ± 7.79	264.0 ± 22.73	56.67 ± 25.62	2.67 ± 1.7	86.33 ± 0.94	36.13 ± 19.29	435.14 ± 28.13	69689
S1	50	50	325.33 ± 2.49	234.0 ± 6.53	85.67 ± 9.84	2.0 ± 0.0	91.0 ± 1.41	81.1 ± 57.32	445.94 ± 65.38	68570
S2	70	30	340.33 ± 2.49	212.0 ± 21.65	116.0 ± 19.6	5.0 ± 2.16	87.33 ± 10.87	32.4 ± 8.36	423.41 ± 31.73	69659
S2	50	50	338.0 ± 4.32	180.33 ± 12.5	144.0 ± 14.31	4.0 ± 1.41	83.33 ± 3.86	153.12 ± 107.98	390.75 ± 23.2	67246
S1	90	-	329.67 ± 0.47	252.33 ± 1.25	65.0 ± 1.41	4.0 ± 2.16	90.0 ± 2.16	178.53 ± 4.38	365.61 ± 24.18	55583
S1	70	-	352.33 ± 12.5	170.33 ± 15.8	128.0 ± 16.97	5.67 ± 2.49	126.0 ± 35.39	100.29 ± 53.06	277.42 ± 13.65	45980
S1	50	-	347.33 ± 4.64	84.33 ± 5.79	184.33 ± 12.68	4.67 ± 2.05	135.0 ± 5.72	192.81 ± 43.39	176.64 ± 22.28	31321

Table 3: Performance of the online version of the LH-MST compared to the offline version of the LH-MST.

performing worst with almost half of its initial seeds unable to shift to a better medoid. It is unsuitable by design as it is biased towards the root by keeping the edges that minimize the distance to the root.

Summary. Compared to the original IS, LH-IS converges to exactly the same medoids in most cases (87% of the clusters); 97% of the medoids depict the same object with only minor viewpoint/color changes. (see supplementary material). For the result in Table 1, the runtime of IS using LH-MST (1 day) is 5 times faster than using MST (5 days). LH-MST is therefore not only a suitable alternative in terms of quality but is also considerably faster than MST.

4.3. Evaluation of online LH-MST

As has been shown in the previous section, LH-MST is suitable for performing IS. Besides the speed advantage in an offline setting, LH-MST also offers the ability to be used with online updates of the image database (Section 3.2).

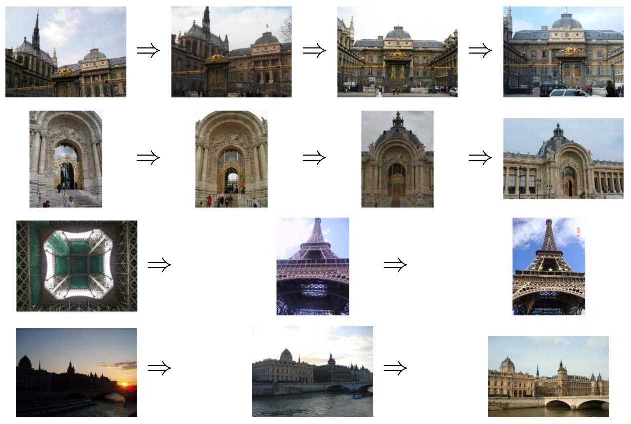


Figure 6: Examples of IS shift sequences using LH-MST.

In the following, we compare the batch with the online version of LH-IS. For each experiment, we split the dataset into initial set and update set. After performing IS using

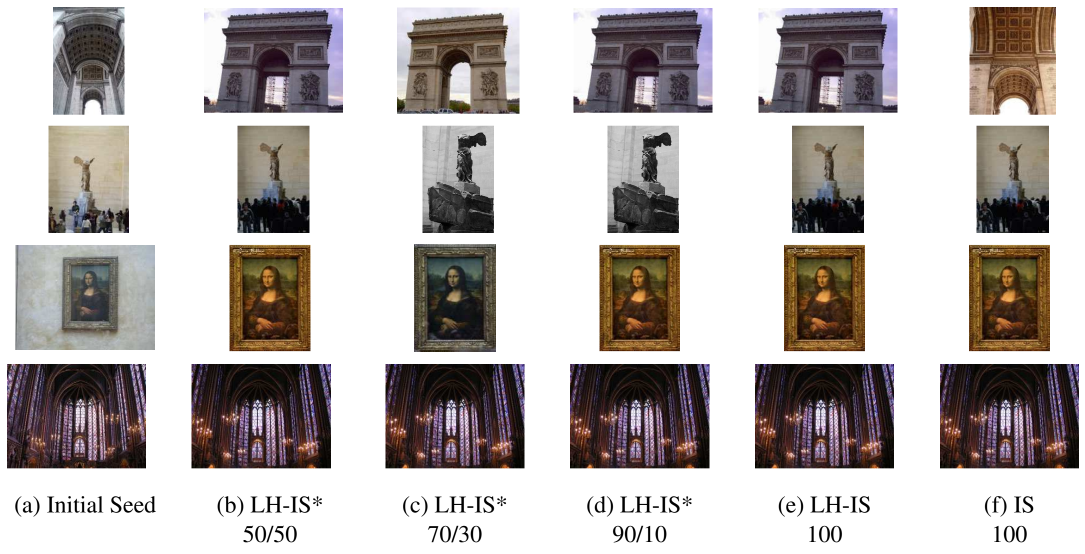


Figure 7: Examples of cluster centers for different version of LH-MST compared to original IS. With * we denote the online version of the algorithm.

LH-MST on this initial set, we add the remaining data via online update. All experiments are repeated 3 times to account for different random splits. To keep the results comparable, we enforce the seed images to be contained in the initial set. Two basic scenarios are considered:

Scenario 1. The split is performed randomly. This scenario mirrors real world setting.

Scenario 2. First, the dataset is split randomly. Second, images that represent medoids in offline LH-IS are shifted into the update set. As such no seed has converged to the "ground truth" cluster prior to the update. This setting gives further insights into the ability of our online version to perform useful shifts during the update step.

Table 3 compares offline LH-IS to various settings of online LH-IS. For both scenarios, we report results using different ratios of images in the initial and update set (from 50/50 to 90/10).

As it can be seen from Table 3 the online version of LH-IS offers comparable results to the offline version. As expected, the selected medoids are closer with a smaller update set. Even in the case of a 50/50 split, many of the resulting medoids agree with offline LH-IS. Note that such a set-up is quite unrealistic considering the drastic change of the database (doubles in size). Yet, in Scenario 1 on average still 234 of the medoids agree with the offline LH-IS compared to 84 if no update is performed. For Scenario 2, although none of the correct medoids were in the initial database ($\mathbb{M}_A = 0$), more than 2/3 of them are recovered through the update (in the 70/30 split) confirming that our update step is able to add meaningful connections in the graph and to also shift medoids to their rightful place.

A detailed comparison between online LH-IS to original IS can be found in the supplementary material.

Figure 7 shows various example medoids resulting from different splits (Scenario 1). In some cases the medoids are different, but visually very similar images are selected (e.g. Mona Lisa in LH-IS, 70/30 split).

Summary. In the realistic scenario of a random initialization of 90% and an update of 10% using online LH-IS, 82% of the medoids are exactly the same as in offline LH-IS; 77% when compared to IS; 98% of the clusters depict the same object. The runtime of the LH-IS update is $\sim$3 hours, 8 times faster than re-computing LH-MST from scratch; 40 times faster than rerunning the original IS algorithm.

5. Conclusion

In this paper we introduced the Limited Horizon Minimum Spanning Tree (LH-MST), an approximation of the well known Minimum Spanning Tree (MST). In contrast to the MST, the LH-MST can be constructed much faster and has the additional property of allowing online updates of the tree structure, even for large changes in the image database. Yet various experiments have shown that this approximation is very close to the original algorithm. Our novel LH-MST can potentially be used in many algorithms which involve a costly Minimum Spanning Tree (MST) calculation. Finally, we demonstrated its applicability for two distinct methods of landmark discovery in large datasets: Single-Link Agglomerative and Iconoid Shift clustering.

Acknowledgements. We thank Tobias Weyand for his invaluable input regarding the IS framework. This work is supported by the Google Faculty Research Award.

References

[1] S. Agarwal, Y. Furukawa, N. Snavely, I. Simon, B. Curless, S. Seitz, and R. Szeliski. Building Rome in a Day. In *ICCV*, 2009.

[2] Y. Avrithis, Y. Kalantidis, G. Tolias, and E. Spyrou. Retrieving landmark and non-landmark images from community photo collections. In *MM*, 2010.

[3] F. Chin and D. Houck. Algorithms for Updating Minimal Spanning Trees. *Journal of Computer and System Sciences*, 16:333–344, 1978.

[4] O. Chum and J. Matas. Web scale image clustering–large scale discovery of spatially related images. *PAMI*, 32(2):371–377, 2010.

[5] O. Chum, J. Philbin, J. Sivic, M. Isard, and A. Zisserman. Total Recall: Automatic Query Expansion with a Generative Feature Model for Object Retrieval. In *ICCV*, 2007.

[6] D. Comaniciu and P. Meer. Mean shift: a robust approach toward feature space analysis. *PAMI*, 24(5):603–619, 2002.

[7] J.-M. Frahm, P. Fite-Georgel, D. Gallup, T. Johnson, R. Raguram, C. Wu, Y.-H. Jen, E. Dunn, B. Clipp, S. Lazebnik, and M. Pollefeys. Building Rome on a Cloudless Day. In *ECCV*, 2010.

[8] S. Gammeter, L. Bossard, T. Quack, and L. Van Gool. I know what you did last summer: object-level auto-annotation of holiday snaps. In *ICCV*, 2009.

[9] K. Heath, N. Gelfand, M. Ovsjanikov, M. Aanjaneya, and L. Guibas. Image webs: Computing and exploiting connectivity in image collections. In *CVPR*, 2010.

[10] Y. Jing and S. Baluja. Pagerank for product image search. In *WWW*, 2008.

[11] D. R. Karger, P. N. Klein, and R. E. Tarjan. A randomized linear-time algorithm to find minimum spanning trees. *J. ACM*, 42(2):321–328, 1995.

[12] G. Kim, C. Faloutsos, and M. Hebert. Unsupervised Modeling of Object Categories Using Link Analysis Techniques. In *CVPR*, 2008.

[13] X. Li, C. Wu, C. Zach, S. Lazebnik, and J.-M. Frahm. Modeling and Recognition of Landmark Image Collections Using Iconic Scene Graphs. In *ECCV*, 2008.

[14] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *CVPR*, 2007.

[15] J. Philbin and A. Zisserman. Object mining using a matching graph on very large image collections. In *ICCVGIP*, 2008.

[16] T. Quack, B. Leibe, and L. Van Gool. World-scale Mining of Objects and Events from Community Photo Collections. In *CIVR*, 2008.

[17] F. Rohlf. Hierarchical Clustering using the Minimum Spanning Tree. *The Computer Journal*, 16:93–95, 1973.

[18] Y. A. Sheikh, E. Khan, and T. Kanade. Mode-seeking by Medoidshifts. In *ICCV*, 2007.

[19] I. Simon, N. Snavely, and S. Seitz. Scene Summarization for Online Image Collections. In *ICCV*, 2007.

[20] P. Spira and A. Pan. On Finding and Updating Spanning Trees and Shortest Paths. *SIAM J. Comput.*, 4(3):375–380, 1975.

[21] T. Weyand, J. Hosang, and B. Leibe. An evaluation of two automatic landmark building discovery algorithms for city reconstruction. In *ECCV Workshops*, 2010.

[22] T. Weyand and B. Leibe. Discovering Favorite Views of Popular Places with Iconoid Shift. In *ICCV*, 2011.

[23] T. Weyand and B. Leibe. Hierarchical Iconoid Shift Discovering Details and Scene Structure with Hierarchical Iconoid Shift. In *ICCV*, 2013.

[24] Y.-T. Zheng, M. Zhao, Y. Song, H. Adam, U. Buddemeier, R. Bissacco, O. Brucher, T.-S. Chua, and H. Neven. Tour the world: Building a web-scale landmark recognition engine. In *CVPR*, 2009.